

An Epics Data Archiver using MySQL, Python, and Apache

Matthew Newville

Consortium for Advanced Radiation Sciences
University of Chicago

October 12, 2010

<http://cars9.uchicago.edu/cgi-bin/pvarch/>

Why an(other) Epics Data Logger?

- Started in 2001, actually. Wanted to learn SQL.
- Use a relational database (MySQL) for data storage.
- Web interface over standard ports and with standard web tools.
 - ▶ get beam-line status information from anywhere
 - ▶ access, view historical data for diagnosing problems.
- Automated Alerts (email) when a condition is met.
- **NOT** fast data collection – that's a different application.

There are many similar Archivers (even at different APS beamlines)

Why an(other) Epics Data Logger?

- Started in 2001, actually. Wanted to learn SQL.
- Use a relational database (MySQL) for data storage.
- Web interface over standard ports and with standard web tools.
 - ▶ get beam-line status information from anywhere
 - ▶ access, view historical data for diagnosing problems.
- Automated Alerts (email) when a condition is met.
- **NOT** fast data collection – that's a different application.

There are many similar Archivers (even at different APS beamlines)

Are Archivers so simple that everyone just builds their own?

Can anything be learned from any one of the implementations?

Is a relational database a good fit for Epics Archival Data?

A tour of the web interface (GSECARS Archiver)

GSECARS Beamline Status: Thu Oct 7 13:31:18 2010 [Settings/Admin Help](#)

[General](#) [APS](#) [ID Neos](#) [ID Water](#) [ID Vacuum](#) [B1 Neos](#) [B1 Water](#) [B1 Vacuum](#) [IDC XRM](#) [BMD](#) [Instruments](#)

Storage Ring

Machine Status [Delivered Beam](#)
Storage Ring Current (mA) 102.072
Storage Ring Lifetime (hours) [7.134](#)
Operating Mode [USER OPERATIONS](#)
ACIS Shutter Permit [PERMIT](#)
13-ID-A Shutter Permit [Enable](#)
Chain A alarm value 0

ID EPS

Front End Valve [Open](#)
Front End Shutter [Open](#)
Vacuum Status [Ok](#)
EPS Status [Ok](#)
White Beam Stop [Closed](#)

BM EPS

Front End Valve [Open](#)
Front End Shutter [Open](#)
EPS Status [Ok](#)
BMC EPS Status [Ok](#)
BMD EPS Status [Ok](#)

Configuration

ID Stations Searched (A,B,C,D) [Yes, Yes, Yes, Yes](#)
BM Stations Searched (A,B,C,D) [Yes, Yes, No, Yes](#)
He gas farm (left, right) [Ok, Ok](#)
Air Temps (13IDD, 13IDC, 13BMD, Roof) (F) [68.104, 62.247, 72.374, 72.941](#)

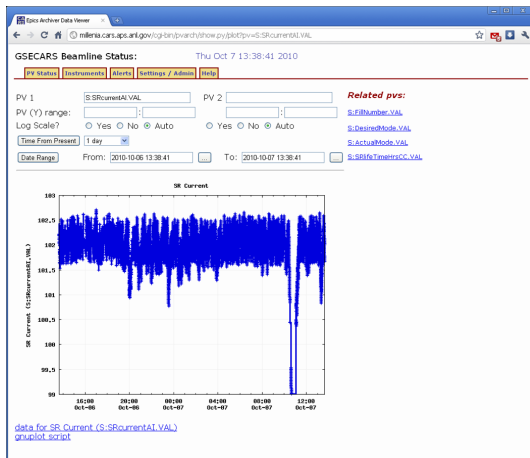
[GSECARS](#) | [Beamline Web Cameras](#) | [APS Storage Ring Status](#) | [APS Facility Page](#) | [APS OAG Data](#)

Main features:

- \gtrsim 5000 PVs
- MySQL
- Apache
- Python
- CA callbacks
- web-definable alerts
- tabs separate PVs by sub-systems
- templates for web pages

PV values displayed as html links to Plot of Data

Plotting Historical Data

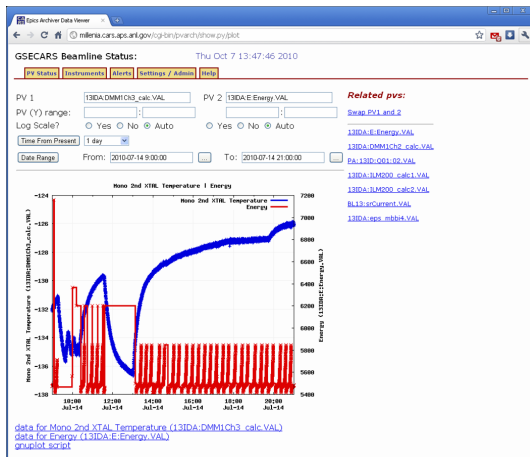


Plots:

- default to past day
- using Gnuplot (currently)
- Plot “From now” or with “Date Range”
- Plot up to 2 PVs
- “Related PVs” list for common pair plots
- pop-up javascript Calendar for Date Range
- String labels for Enum PVs

Data file and plotting script available as a download

Plotting Historical Data

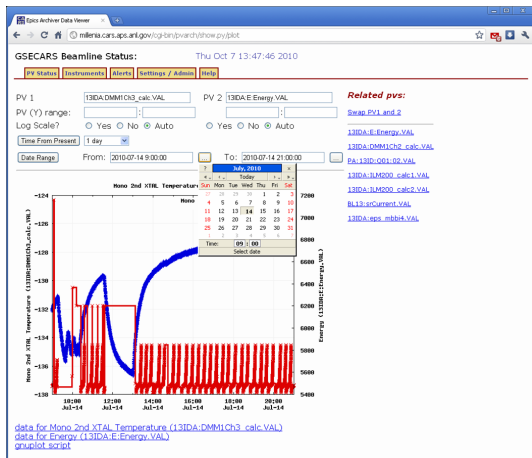


Plots:

- default to past day
- using Gnuplot (currently)
- Plot "From now" or with "Date Range"
- Plot up to 2 PVs
- "Related PVs" list for common pair plots
- pop-up javascript Calendar for Date Range
- String labels for Enum PVs

Data file and plotting script available as a download

Plotting Historical Data

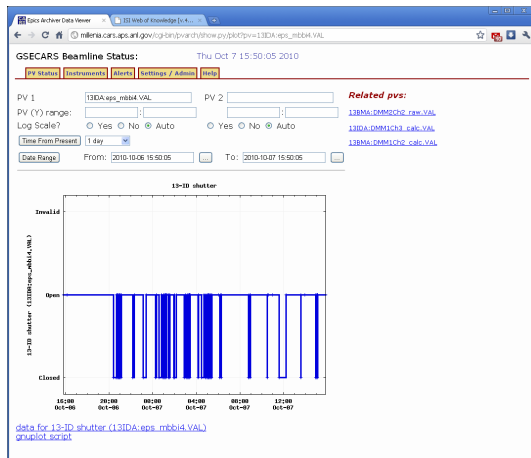


Plots:

- default to past day
- using Gnuplot (currently)
- Plot "From now" or with "Date Range"
- Plot up to 2 PVs
- "Related PVs" list for common pair plots
- pop-up javascript Calendar for Date Range
- String labels for Enum PVs

Data file and plotting script available as a download

Plotting Historical Data



Plots:

- default to past day
- using Gnuplot (currently)
- Plot "From now" or with "Date Range"
- Plot up to 2 PVs
- "Related PVs" list for common pair plots
- pop-up javascript Calendar for Date Range
- String labels for Enum PVs

Data file and plotting script available as a download

Templates for Web Status Pages

The web status pages are generated from a simple template:

Web Page Template

```
[Storage Ring]
S:ActualMode.VAL      | Machine Status
S:SRcurrentAI.VAL     | Storage Ring Current (mA) | %8.3f
S:SRlifeTimeHrsCC.VAL | Storage Ring Lifetime (hours)| %8.3f
S:DesiredMode.VAL     | Operating Mode
...

[ID EPS]
13IDA:eps_mbbi57.VAL  | Front End Valve
13IDA:eps_mbbi4.VAL   | Front End Shutter
13IDA:eps_mbbi81.VAL  | Vacuum Status
...

[Configuration]
PA:13ID:Q01:00.VAL,...| ID Stations Searched (A,B) | yes/no
PA:13BM:Q01:00.VAL,...| BM Stations Searched (A,B) | yes/no
...
```

GSECARS Beamline Status: Thu Oct 7 13:31:18 2010 [Settings/Admin Help](#)

[General](#) [APS](#) [ID Neutrons](#) [ID Water](#) [ID Vacuum](#) [ID Neutrons](#) [ID Water](#) [ID Vacuum](#) [ID XRD](#) [ID](#) [Instruments](#)

Storage Ring

Machine Status	Delivered Beam
Storage Ring Current (mA)	102.077
Storage Ring Lifetime (hours)	7.134
Operating Mode	USER OPERATIONS
ACTIS Shutter Permit	PERMIT
13-ID-A Shutter Permit	Enable
Chain A alarm value	0

ID EPS

Front End Valve	Open
Front End Shutter	Open
Vacuum Status	Ok
EPS Status	Ok
White Beam Stop	Closed

BM EPS

Front End Valve	Open
Front End Shutter	Open
EPS Status	Ok
BMC EPS Status	Ok
BMD EPS Status	Ok

Configuration

ID Stations Searched (A,B,C,D)	Yes, Yes, Yes, Yes
BM Stations Searched (A,B,C,D)	Yes, Yes, No, Yes
He gas farm (left, right)	Ok, Ok
Air Temps (13IDD, 13IDC, 13BMD, Roof) (F)	68.104, 62.247, 72.374, 72.941

[[GSECARS](#) | [Beamline Web Cameras](#) | [APS Storage Ring Status](#) | [APS Facility Page](#) | [APS OAG Data](#)]

Each “Tab” has a separate template file.

Templates are read dynamically – easy to add and change what is displayed.

A Makefile sets the order of the Tabs.

Main Implementation Issues:

- ① Epics CA needs a connection for each PV for each process / thread.
- ② A Web server may render each page as a separate process.
- ③ The Archiving needs continual access to new values for all PVs.

A web interface could be very slow, waiting for PV connections even though most will disappear and be reconnect for the next page.

Architecture: A Cache of Current PV values

Main Implementation Issues:

- ① Epics CA needs a connection for each PV for each process / thread.
- ② A Web server may render each page as a separate process.
- ③ The Archiving needs continual access to new values for all PVs.

A web interface could be very slow, waiting for PV connections even though most will disappear and be reconnect for the next page.

Key Implementation Decision: Break Archive task into two processes.

Caching Process stays connected to all PVs, maintains table of latest values using CA callbacks, handles alerts.

Archiving Process reads PV data for Cache table, archives as needed.

Web pages simply access the Cache tables to read the latest PV values.
1 DB connection is much faster than 100 PV connections.

The Cache table looks like this:

Cache Table (SQL)

```
create table cache (  
  id          int unsigned not null primary key auto_increment,  
  pvname      varchar(64) not null,  
  type        varchar(64) not null default 'int',  
  value       tinyblob   default null,  
  cvalue      varchar(64) default null,  
  ts          double not null default 0,  
  active      enum('yes','no') not null default 'yes');  
create index pvname_id on cache (pvname);
```

The Cache table holds current data:
(Name, Value, StringValue, Time)

These are kept up-to-date with
CA callbacks.
SQL UPDATE

The Cache table looks like this:

Cache Table (SQL)

```
create table cache (  
  id          int unsigned not null primary key auto_increment,  
  pvname      varchar(64) not null,  
  type        varchar(64) not null default 'int',  
  value       tinyblob   default null,  
  cvalue      varchar(64) default null,  
  ts          double not null default 0,  
  active      enum('yes','no') not null default 'yes');  
create index pvname_id on cache (pvname);
```

The Cache table holds current data:
(Name, Value, StringValue, Time)

These are kept up-to-date with
CA callbacks.
SQL UPDATE

What gets Cached? What gets Archived?

5000 PVs, 80% from Epics Motors. 436 Motors, each with 10 fields:

.VAL .OFF .FOFF .SET .HLS .LLS .DIR _able.VAL .SPMG .DESC

are monitored and archived. These rarely change. But we want to know when they do.

~ 600 other variables. Very few arrays (only CHAR waveforms used as long strings).

Archiving Process and databases

The **pvarch** program:

- allows adding, deleting PVs.
- starts and stop caching and archiving process.
- organizes archival data into “Runs” (~monthly).

Each **Run** pvarchive_00001, pvarchive_00002, has these tables:

Table	Important Columns	Column	Meaning
PV	ID, Name, DataTable, DeadTime, DeadBand	DataTable	which pvdatNNN data is stored in.
pvdat001	PV_ID, TimeStamp, Value	DeadTime	time to wait between archive writes.
pvdat002	PV_ID, TimeStamp, Value	DeadBand	fractional change in value to ignore.
.			
.			
pvdat128	PV_ID, TimeStamp, Value		

The data stores were broken up into 128 tables for faster lookups.

This should be re-evaluated.

The Archiving Process:

- reads recent data from the Cache, writes to the Archive tables.
 - *throttles* recording of changes with a DeadTime and DeadBand.
-

Throttling with DeadTime and DeadBand

DeadTime:

After recording a change, further changes are not recorded until the *DeadTime* has expired.

If many new value arrives in that interval, only the final value will be recorded.

Typical DeadTime = 1 sec.

DeadBand:

minimum fractional change recorded for DOUBLE/FLOAT PVs.

Architecture: Other Tables

Other Tables hold information for

- Runs** lists Archives database by time range of “Run”.

- Pair Scores** for Related Pairs of Variables.

- Instruments** groups of PVs to treat as a logical unit.

- Alerts** data for email alerts on alarm conditions

Architecture: Other Tables

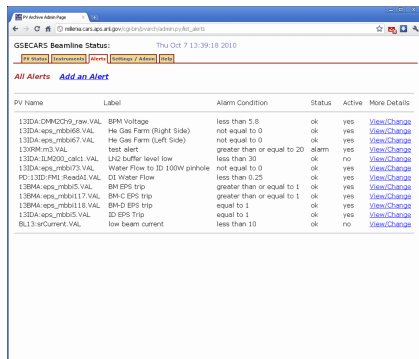
Other Tables hold information for

Runs lists Archives database by time range of "Run".

Pair Scores for Related Pairs of Variables.

Instruments groups of PVs to treat as a logical unit.

Alerts data for email alerts on alarm conditions

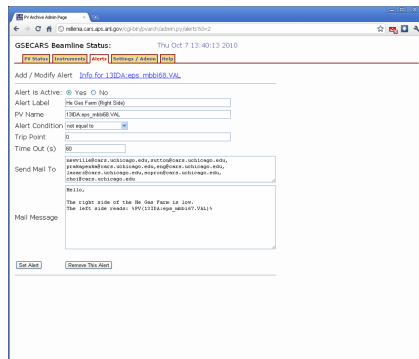


GSECARS Beamline Status: Thu Oct 7 13:39:18 2010

[PV Status](#) [Instruments](#) [Alerts](#) [Settings / Admin](#) [Help](#)

All Alerts [Add an Alert](#)

PV Name	Label	Alarm Condition	Status	Active	More Details
13IDA:DMMDCH9_row.VAL	BPM Voltage	less than 5.8	ok	yes	View/Change
13IDA:eps_mbbi68.VAL	He Gas Farm (Right Side)	not equal to 0	ok	yes	View/Change
13IDA:eps_mbbi67.VAL	He Gas Farm (Left Side)	not equal to 0	ok	yes	View/Change
13ORRM3.VAL	test alert	greater than or equal to 20	alarm	yes	View/Change
13IDA:ILM200_csdct.VAL	LN2 buffer level low	less than 30	ok	no	View/Change
13IDA:eps_mbbi73.VAL	Water Flow to ID 100W pinhole	not equal to 0	ok	yes	View/Change
PO-13ID-FML-ReadA1.VAL	DI Water Flow	less than 0.25	ok	yes	View/Change
13BMA:eps_mbbi5.VAL	BM EPS trip	greater than or equal to 1	ok	yes	View/Change
13BMA:eps_mbbi117.VAL	BM-C EPS trip	greater than or equal to 1	ok	yes	View/Change
13BMA:eps_mbbi118.VAL	BM-D EPS trip	equal to 1	ok	yes	View/Change
13IDA:eps_mbbi5.VAL	ID EPS Trip	equal to 1	ok	yes	View/Change
BL13-scCurrent.VAL	low beam current	less than 10	ok	no	View/Change



GSECARS Beamline Status: Thu Oct 7 13:40:13 2010

[PV Status](#) [Instruments](#) [Alerts](#) [Settings / Admin](#) [Help](#)

[Add / Modify Alert](#) [Info for 13IDA:eps_mbbi68.VAL](#)

Alert is Active: ☒ Yes ☐ No

Alert Label:

PV Name:

Alert Condition:

Trip Point:

Time Out (s):

Send Mail To:

Mail Message:

[Get Alert](#) [Remove This Alert](#)

Cache Startup Statistics:

#	Message	Delta(s)	Total(s)
	connecting to 5028 PVs	0.238	0.238
	Created 5028 PV Objects	6.430	6.668
	Connected to PVs (60 not connected)	29.063	35.730
	got initial values for PVs	7.597	43.327
	Entered values for 5028 PVs to Db	0.939	44.266
	added callbacks for PVs	0.035	44.301
	looked for unconnected pvs: 60 not connected	5.867	50.168
	pvs connected, ready to run. Cache Process ID= 21317		

Starting Cache process for
~5000 PVs takes ~50 sec
includes getting DBR_CTRL
data and looking twice for
un-connected PVs

Performance: Caching

Cache Startup Statistics:

#	Message	Delta(s)	Total(s)
	connecting to 5028 PVs	0.238	0.238
	Created 5028 PV Objects	6.430	6.668
	Connected to PVs (60 not connected)	29.063	35.730
	got initial values for PVs	7.597	43.327
	Entered values for 5028 PVs to Db	0.939	44.266
	added callbacks for PVs	0.035	44.301
	looked for unconnected pvs: 60 not connected	5.867	50.168
	pvs connected, ready to run. Cache Process ID= 21317		

Starting Cache process for
~5000 PVs takes ~50 sec
includes getting DBR_CTRL
data and looking twice for
un-connected PVs

Cache Loop:

```
def onChanges(self, pvname=None, value=None, char_value=None,
               timestamp=None, **kw):
    self.data[pvname] = (value, char_value, timestamp)
    if pvname in self.alert_data:
        self.alert_data[pvname]['last_value'].append(value)

def mainloop():
    connect_to_db()
    connect_to_PVs()
    while True:
        epics.poll(1.e-4)
        update_cache_table()
        every_15_seconds:
            process_alerts()
```

The Cache loop runs at
~150 Hz (~7 msec/loop)

Most changes are from ~100
PVs that change at ~10Hz.

Alerts are handled here.

Performance: More Statistics (5028 PVs)

Hardware: 2 Dual-Core Opterons 2.4GHz, 8Gb RAM, SCSI disks.

Process	Activity	Typical Value
Cache	Pend/Update Loops per second	120
	Total Updates per second	150
	PVs changed in past minute	100
	PVs changed in past hour	200
Archive	Loops per second	50
	Values Archived per second	20

Typical top output:

```
Cpu0 : 3.7%us, 3.0%sy, 0.0%ni, 47.0%id, 43.0%wa, 1.7%hi, 1.7%si, 0.0%st
Cpu1 : 14.0%us, 2.0%sy, 0.0%ni, 84.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu2 : 1.3%us, 0.0%sy, 0.0%ni, 98.7%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Cpu3 : 54.0%us, 1.0%sy, 0.0%ni, 45.0%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem:  8264184k total, 4782832k used, 3481352k free, 405116k buffers
Swap: 5406712k total, 49344k used, 5357368k free, 3234884k cached

  PID USER      VIRT  RES  SHR  S  %CPU  %MEM   TIME+  COMMAND
15958 mysql      750m 573m 6284 S  51.5   7.1  2482:49 /usr/libexec/mysqld ...
18823 root       60436 27m 2576 S  13.9   0.3  486:22.26 python /usr/bin/pvarch start
21317 root       77972 34m 3548 S  10.6   0.4  401:56.85 python /usr/bin/pvarch cache start
```

System archives ~15 Mb / day.

Is Epics Archival data relational?

Maybe. A relational DB is really good for asking:

Show me the Mirror Angles when the Ring Current < 100 mA and Monochromator Temperature $> -120^{\circ}$ C since last April.

I **never** do this. For me, the data has one axis: *Time*.

- 1 Get Mirror Angle, Ring Current, and Monochromator Temperature since last April.
- 2 Filter data based on Ring Current and Monochromator Temperature.

Is Epics Archival data relational?

Maybe. A relational DB is really good for asking:

Show me the Mirror Angles when the Ring Current < 100 mA and Monochromator Temperature $> -120^{\circ}$ C since last April.

I **never** do this. For me, the data has one axis: *Time*.

- 1 Get Mirror Angle, Ring Current, and Monochromator Temperature since last April.
- 2 Filter data based on Ring Current and Monochromator Temperature.

Maybe I'm just too lazy to write complex SELECT statements.

But: An RDBMS has many other useful features built in:

- Data integrity and portability.
- Scales to huge data sizes.
- Client-Server model for multiple data writers and readers.
- A security model.
- Years of work behind it.

Archival data should be stored in an SQL-based RDBMS unless this can be shown to not work.

The web interface matters.

Ease of configuration and use matters.

For high performance and scale, use multiple processes, data stores, and server processes.

<http://cars9.uchicago.edu/cgi-bin/pvarch/>